High Res Mock Driver

Goals

1. Understand the robot
2. Understand the attached API
3. Understand the description
4. Think of an efficient design
5. Pro/Cons of my solution

Understand the robot

1. Robots functionality
    a. Robotic arm that moves objects around various locations in a system
    b. Remote controlled
    c. Starts on socket 1000
2. Robots input
    a. A string thats split into 2 parts with a %
        i. Command name
        ii. %
        iii. Parameters for command
3. Robots rules
    a. 1 process at a time

Understand the attached API

1. Has four commands the program can utilize
    a. public int home()
        i. return processID
    b. public int pick(int sourceLocation)
        i. return processID
    c. public int place(int destinationLocation)
        i. return processID
    d. public String status(int processID)
        i. return processStatus
2. Description
    a. home()
        i. The robot begins its “homing process” which needs to complete before the robot moves the samples. Returns a unique process id which can track the process of homing.

            1. **Assumption - Homing process is the process of returning the robot to its original state in order to prepare it for a new command.**
    b. pick()
        i. Tells the robot to begin a process of picking up a sample from a location. The onboard software gives a unique id to track this process
    c. place()
        i. Tells the robot to place a sample to a location. Returns a unique id to track this process.
    d. status()
        i. This just gives the the status of the supplied process. Can return 3 String values
            1. In Progress
            2. Finished Successfully
            3. Terminated with Error

Understand the description

1. There is a UI interface that allows the user to interact with the robot API. The user will type out a parameter (if applicable) and then press the button in order to execute the command. The UI will then return either an empty string if the operation is successful or a string with a description of an error that occurred during the function.
2. Interface Methods
    a. OpenConnection(String IPAddress)
        i. Establishes the initial connection with the MockRobot onboard software
        ii. IPAddress is the address of the specific robot you are trying to connect to
    b. Initialize()
        i. Initialize sets the robot into the home() state. This prepares it to do any other commands after.
    c. ExecuteOperation(String Operation, String[] parameterNames, String parameterValues)
        i. String operation
            1. Pick
            2. Place
            3. Transfer
        ii. String[] parameterNames
            1. an array that has the name of each parameter to be used for the given operation
        iii. String[] parameterValues
            1. An array that contains the value of each parameter to be used for the given operation.
        iv. Quick Notes

1. The two arrays are parallel

d. Abort()
    i. Upon clicking this button, all communication with the robot is terminated

3. Operation
    a. One Initialization
    b. One Connection
    c. N number of operations
    d. Aborting after any Nth operation is finished

Think of an efficient design

1. Server
    a. Step 1: Initialization
        i. My server will represent the robot software. The server will have the api installed within it. The server upon connection will be waiting for the client to send 3 values, a string with an “initialization” command followed by 2 null arrays. (this ensures that the client sets the robot to a state from which it can execute commands.
    b. Step 2: Operation
        i. After this initialization step is completed, the robot will wait for 3 values again, this time it will want the arrays to be populated, if not, it returns a string to the user stating “Could not complete operation, no values received”. This will ensure the user knows they need to have values and is a failsafe in case they somehow bypass the UI validation process, this message should never actually come up.
    c. Step 3: Execution
        i. Create a null String variable named “response”
        ii. While there is more operations, we keep on looping through one of the arrays, this is more efficient than using .length since we shorten our runtime by order of N.
        iii. Operation Type
            1. Pick
                a. This operation will call upon the Pick() method in the API, what this will accomplish is the server will save the string values in 2 variables which are global.
                    i. String Name
                    ii. Value Value
                b. Here are some of the use cases:
                    i. Success
                        1. The value and name exists, we save them to the variables.
                        2. response = Finished Successfully

ii. Failure 1
    1. The value exists but the name does not, we return an error to the user stating that the arrays are inconsistent
    2. response = Error: arrays inconsistent
iii. Failure 2
    1. The name exists but the value does not, we return an error to the user stating that the arrays are inconsistent
    2. response = Error: arrays inconsistent
iv. Failure 3
    1. Neither the value nor the name exist, this should never be the case but in case it is we return that both are not present.
    2. response = Error: no values present in arrays

2. Place
    a. This operation will call upon the Place() method in the API, what this will accomplish is the server will look to see if there are any values stored in the global variables and then attempt to place the items in the location.
    b. If no variables are stored, the response message becomes “Error: Nothing has been picked up”
    c. If the slots in the array are filled already, the response becomes “Do you wish to overwrite the current values (a:b) in the destination for place operation (x:y) where x is the name and y is the value to which the user can reply “yes, y, no, n” and case does not matter.
        i. Here are some of the use cases:
            1. Success 1
                a. The value and name are placed in the correct location without a hiccup.
                b. response = Finished Successfully
            2. Success 2
                a. The value and the name exist but the location is already populated, the user must decide whether they want to override or not.
                b. first response = Do you wish to override?
                c. second response = Finished Successfully
            3. Failure 1

            a. No global values are present
            b. response = Error: Nothing was picked up
        3. Transfer
            a. This will just run the commands Pick and Place back to back with similar lines of success and failure.
            b. We are assuming that when a transfer operation is called, the arrays have at least 2 values. This will just swap their locations.
    d. Step 4: Response
        i. Whatever the response message we came to through the last few steps, it will give it to the user.
2. Client
    a. The client has a screen where they can choose from 4 options
        i. Open Connection
        ii. Initialize
        iii. Execute Operation
        iv. Abort
    b. If the client has not yet Opened Connection, they must choose the first option, if they choose any other option (besides abort), they will receive an error stating they have not established a connection yet.
    c. After establishing a connection, the client must Initialize, if the client attempts to execute an operation before initializing, they will receive an error stating the robot has not yet been initialized
        i. The client at this step can alternatively close the existing connection with abort() or open a new connection which will close the current connection which will loop them back to the initialization step
    d. The client can abort at any point which will close the connection and end the program.

Pro/Cons of my solution

1. Pro
    a. The information is really client heavy thus forming an asymmetric connection which allows the robot to not be overburdened.
    b. I allow the client to be a little flexible while keeping them on rails in order to minimize errors
2. Con
    a. There isn't a lot of freedom, this robot will only be good at picking things up and putting them down

b. I'm sure this isn't the most efficient way since I am consistently sending information through the network when instead I could possibly enable some sort of queue for the robot.